IRREGULAR MESH AND EMBEDDED GEOMETRIC DESCRIPTION IN A COMPUTER GRAPHICS SYSTEM

FIELD OF THE INVENTION

The present invention relates generally to computer graphics, and topological and geometrical representation of 3-dimensional objects. More particularly the invention relates to description of objects using meshes.

BACKGROUND OF THE INVENTION

Representations of 3-dimensional objects in computer graphics are often based on the creation of some kind of mesh consisting of vertices connected by edges and faces. This mesh creates a frame upon which surfaces and texture can be superimposed. Such a model has two distinct, but interrelated, properties, namely the topology, which is a description of how the various vertices are connected by edges and faces, and the geometry, which is the position in space of the vertices and the shape of the surface superimposed on the mesh.

Boundary based representation of objects (B-rep) has been exploited for a number of years within such fields as Computer Aided Design (CAD), modeling of geological structures, and entertainment such as games and movies. B-reps represent objects by their boundaries. For example, a 3D cube is represented by its six faces, where the faces are tied together with some relationship information. An object is usually subdivided further, such that volumes, faces, edges and vertices are represented explicitly, along with the positional relationships between these entities. The various B-rep based topological representations differ in their level of subdivision, the relationships established between the topological elements, and how they distinguish between the topological model and the data embedded in this model, e.g. geometry.

Subdivision is a technique wherein a mesh is successively refined by repeated subdivision. A smooth surface is defined as the limit of a sequence of such refinements. Subdivision allows arbitrary topology, but the topology has not been explicitly defined in prior art solutions. One example of how topology has been handled is in the form of lists, and in order to e.g. find the neighbors of a given face in the mesh, it would be necessary to traverse the entire lists.

An alternative approach has been to use a regular mesh and associate a spline patch with each face. A refinement of this method, known as hierarchical B-splines, involves refining the mesh locally and adding smaller patches to the refined area. Again, this method lacks any explicit data model representing the topology.

Various representations can be chosen for the geometric entities embedded in the topological model. At a simplest level, straight lines are used to describe curves

between vertices, and triangulation is used to create surfaces. However, it is desirable to represent free-form surfaces. A well-known way of doing this is to use surface splines as described e.g. in Jörg Peters, Biquadratic C^1-surface splines over irregular meshes, Computer Aided Design Volume 27 Number 12, December 1995, pp. 895-903, 1995.

A B-spline is a smooth free-form surface defined by a set of mxn control vertices in a regular mesh. A complex surface can be constructed by creating a mesh of control vertices. In order to reduce the number of control vertices necessary for representing a surface with a number of local changes or deformations, a method of using hierarchical B-splines is known from David Forsey, Hierarchical B-splines, published on the Internet at http://www.cs.ubc.ca/nest/imager/contributions/forsey/dragon/hbsplines.html. This technique consists of adding a B-spline patch to the region around the deformation, necessitating the addition of control vertices only to this patch.

Bruno Lévy and Jean-Laurent Mallet, Cellular Modeling in Arbitrary Dimension using Generalized Maps, pp 1-11, 05.01.2001, describes how Generalized Maps, which will be described below, can be used to represent the topology of a geometric model also containing a geometrical description.

Veronique Lang and Pascal Lienhardt, Simplical Sets and Triangular Patches, Proceedings of Computer Graphics International, pp 154-163, 24.06.1996, combines simplical sets, a subset of Generalized Maps, with triangular Bezier patches.

SUMMARY OF THE INVENTION

Common to the prior art solutions is that they do not provide an effective data structure that allows efficient interaction between objects, and dynamic change of topology and geometry as a result of such interactions. The present invention aims at providing a method and a data structure that facilitates fast and efficient local interaction between 3D-objects, and fast and efficient local refinement or local deformation of the 3D-object descriptions.

The present invention is based on the use of generalized maps (G-maps) in order to describe the mesh topology, and on a smooth geometric description of the surface superimposed on this mesh.

Generalized maps (G-maps) is a type of generalized boundary representation model where the basic topological element of the topological map is the dart, a semi edge of a graph. A map is defined by a set of darts, D, and a set of involutions (α_0, α_1, ..., α_n). Each involution "α_i" describes the links between darts corresponding to a dimension "i". In 3-dimensional space, these relationships are used to subdivide an

object into topological volumes (α_3), a volume into topological faces (α_2), a face into topological edges (α_1), and an edge into topological vertices (α_0).

Spatial representation of a 3-dimensional topological map consists in associating geometric entities with cells. Points in space are associated with vertices (0-cells), curve arcs are associated with edges (1-cells), surfaces are associated with faces (2-cells), and geometric volumes are associated with topological volumes (3-cells). These associations constitute the embedding of geometrical data in the topological map.

The n-dimensional G-map is expressed as

$G = (D, \alpha_0, \alpha_1, \ldots, \alpha_n)$

where D is a finite set of darts, and $\alpha_0 \ldots \alpha_n$ are n+1 involutions on D. An involution is a set of darts couples and/or singles. In the 3-dimensional case, the darts are used as follows:

α_0 creates edges by linking darts of adjacent vertices;
α_1 connects face edges by linking darts that meet in common vertices;
α_2 connects surfaces by linking darts that constitute a common edge;
α_3 connects volumes along a common face by linking darts that constitute the common face.

According to a first aspect of the invention, there is provided a method for describing a 3D-object using a mesh based geometric model where the topology of the description is described using G-maps and the geometry is based on coordinates in space associated with the vertices of said mesh.

One way of creating the geometry description is to repeatedly apply a subdivision algorithm until a sufficiently smooth surface has been applied.

For an example of subdivision algorithms, reference is made to E. Catmull and J. Clark, Recursively generated B-spline surfaces on arbitrary topological surfaces, *Computer-Aided Design* 10(6):350-355, November 1978.

A preferred way of creating the geometric description, however, is based on the creation of a refined inner mesh by applying a subdivision algorithm once, and using the vertices of this second mesh for the creation of surface patches associated with the first mesh.

In a preferred embodiment of the invention, this is done by dividing each face into quads by defining a point in the middle of the face and defining points that divide each edge in two. Drawing lines from the new face point and down to each new edge point will subdivide the face into only rectangular quads. A smooth surface can then be created e.g. by associating a Bezier-spline patch with each quad.

In order to create local refinement or distortion of the described geometric model, each quad can be subdivided further. According to the invention this is handled by creating a new G-map description for the topology of the subdivided quads and linking this to the previous level in a hierarchical manner. Additional geometric patches are then associated with the quads of the new topological level in the same manner as the original patches were associated with the quads of the original topological level. The invention allows for any number of refinement levels to be added on top of each other hierarchically.

It should be noted that in the above description the difference between topology and geometry is not made explicit. The faces are topological objects, and thus they do not have any geometrical midpoint. To the extent that geometrical coordinates are calculated they are stored in a geometrical model embedded in the topological model.

The various references between objects in the data structure can be implemented in a number of ways. According to a preferred embodiment, two darts that are linked by an α_1 involution will define one corner of a quad, and reference to that quad will then be made through these darts. In the same way, reference from one hierarchical level of the topology to another is made through darts linked by a γ link. Since the topological levels of this hierarchy will represent finer or coarser mesh structures, the reference from a dart on one level to a dart on a finer level will be referred to as $\gamma_{\_fine}$, while a reference from a dart on a finer level to a dart on a coarser level will be referred to as $\gamma_{\_coarse}$. It should be realized that while the α involutions allows navigation through one G-map structure, a γ link results in a move from a G-map on one hierarchical level to a G-map on a higher or lower hierarchical level.

In this manner it will be easy to navigate through the topological structure simply by applying the various α-involutions and check references to quads and/or their corresponding geometrical patches.

In addition to the method of creating a description of the topological and geometrical structure, the invention also covers the data structure resulting from this method.

Another aspect of the invention is a computer system programmed in a manner suitable for generating or changing a description or a data structure according to the invention based on suitable input data representing 3-dimensional objects.

Yet another aspect of the invention is a computer program product which, when installed on a suitable computer, enables the computer to generate a description or a data structure according to the invention, based on suitable input data representing 3-dimensional objects.

DESCRIPTION OF THE DRAWINGS

A preferred embodiment of the invention will now be described with reference to the drawings, where:

Figure 1	is an illustration of the subdivision of a model into cells;
Figure 2	is an illustration of a 2G-map and corresponding topology and possible geometry;

DETAILED DESCRIPTION OF THE INVENTION

The present invention is based on the use of generalized maps (G-maps) applied to the field of computer graphics. The theoretical structure of a G-map forms the basis for a description of a topological mesh which is associated with an embedded geometry. The following description will introduce the G-map structure and give examples of how this structure can be implemented as a data structure in a computer graphics system. These are intended as non-limiting examples, and those skilled in the art will realize that implementation details regarding the actual methods and data structures are possible and are intended to be covered within the scope and spirit of the invention.

The invention is particularly suited to represent 3D models in a computer graphics system where the computer system includes at least one processor which in combination with computer program code means and the various storage and graphics means of the computer is capable of constructing the model based on input data received over an input interface of the computer. The input data may include at least topological data representing vertices and faces of a mesh and geometrical data representing coordinates to be associated with the vertices, as described in further detail below. In a preferred embodiment of the invention the processing means of the computer system in combination with computer program code will constitute the various means of a system operating in accordance with the invention.

FIG. 1 illustrates how a 3D model can be divided into volumes, faces, edges and vertices. A 3D model 101 is subdivided into two volumes 102. These are again subdivided, each into six faces 103, representing each side of the two volumes 102. Then each face is subdivided into four edges 104. Finally, each edge is subdivided into two half-segments, each of which will be referred to as a dart 105. Each dart is associated with exactly one vertex, one edge, one face and one volume, also referred to as 0-cells, 1-cells, 2-cells and 3-cells respectively. It should, however, be noted that it is not necessary to represent all these cells explicitly in the data structure describing the G-map topology. As described below, the preferred embodiment of the invention does not include an explicit representation of edges. Edges are found by using functions operating on the darts. Likewise, the preferred embodiment described herein does not include explicit representations of volumes. The invention does not, however, prevent such descriptions from being included.

A generalized map of dimension $n \geq 0$ is a graph $G = (D, \alpha_0, \alpha_1, \ldots, \alpha_n)$, that consists of a non-empty, finite set of darts, D, and n+1 involutions α_i on D. The involutions create associations between darts in a manner that define the topology of the graph. A more detailed description is given in Y. Halbwachs and Ø. Hjelle, Generalized maps in geological modeling: Object-oriented design of topological kernels, In H. P. Langtangen and A. M. Bruaset and E. Quak, editors, Advances in Software Tools for Scientific Computing, pages 339 - 356, Springer, 1999.

A G-map may in principle have any number of dimensions, but in computer graphics the most relevant is the 3-dimensional case, and the following exemplary description will be limited to that. A person skilled in the art will realize that the invention may be modified to other numbers of dimensions if that should prove convenient as a design option.

FIG. 2 illustrates how the topology of three faces 201, 202, 203 is described using G-maps. The G-map, G, is a set of darts D and involutions α_i, in this case

$G = 2G\text{-map}(D, \alpha_0, \alpha_1, \alpha_2)$,

$D = \{ 1, 2, 3, ..., 24 \}$

$\alpha_0 = \{ (1, 2), (3, 4), (5, 6), (7, 8), (9, 10), (11, 12), (13, 14), (15, 16), (17, 18), (19, 20), (21, 22), (23, 24) \}$

$\alpha_1 = \{ (1, 10), (2, 3), (4, 5), (6, 7), (8, 9), (11, 18), (12, 13), (14, 15), (16, 17), (19, 24), (20, 21), (22, 23) \}$

$\alpha_2 = \{ (1, 18), (2, 17), (3), (4), (5), (6), (7), (8), (9, 20), (10, 19), (11, 24), (12, 23), (13), (14), (15), (16), (19), (21), (22) \}$.

The involutions are sets of couples and singles. A couple (d_i, d_j) in the set α_k is equivalent to $\alpha_k(d_i) = d_j$ and $\alpha_k(d_j) = d_i$, while a single (d) is equivalent to $\alpha_k(d) = d$.

As can be seen from FIG. 2, couples belonging to the α_0 involution join together two darts belonging to different vertices. Such a couple of darts then define an edge between two vertices of the graph. It can also be noted that since each dart is associated with only one face, the edge between two adjacent faces will be implicitly defined twice, except at the edge of a surface, where there is no bordering face.

Couples belonging to the α_1 involution join together two darts belonging to the same vertex. This is where two face edges are joined together. It will be realized that by repeatedly invoking α_0 and α_1 involutions, it is possible to travel along the boundary of a face. A single in the α_1 set would represent an extreme vertex that defined the end of a line or edge.

Couples belonging to the α_2 involution join darts that belong to the same edge but to different faces. These couples define the relationship between adjacent faces. Invoking an α_2 involution makes it possible to go from one face to its neighbor. Singles in the α_2 set indicates that there are edges that do not separate two faces, in other words, edges along the boundary of the mesh. It will be realized that e.g. a 2-G-map representation of a sphere as connected triangles will not have any singles.

It would be possible, but not necessary, for a 3D representation, to include α_3 involutions in order to represent adjacent volumes. The couples belonging to the α_3 set would then represent darts that belong to the same face, but adjacent volumes. Singles in the α_3 set would represent darts around the boundary of faces that represent the surface of the 3D model. Returning to FIG. 1, the two volumes 102 would be joined together by α_3 couples along the edges of the two surfaces facing each other.

An orbit of a dart d is the set of all darts in a G-map that can be reached by applying a subset of the involutions in the G-map, starting with d. It has already been mentioned that repeated application of α_0 and α_1 will define the boundary of a face. This is referred to as the 2-orbit. For 2-G-maps, the 0-orbit (α_1, α_2) is the set of darts incident to a vertex, the 1-orbit (α_0, α_2) is the set of darts incident to an edge, and the 2-orbit (α_0, α_1) is the set of darts incident to a face.

Also shown is the topological representation 204 of the G-map, and a possible geometry 205. The geometry depends on the embedded geometric description, as described below:

FIG. 3 is another example of a simple volume with a topology that is described using G-maps. The topology comprises vertices 301, edges 302 and faces 303. In this example, the topology of the volume is that of a cube 304, while it is illustrated that the geometry could for instance actually be that of a sphere 305. Again it is shown how α_0 involutions link darts together to create the edges, α_1 involutions link edges together at each vertex of the mesh, and α_2 involutions tie various faces together by linking darts that belong to the same edges but adjacent faces. It should be noted that it would be possible to leave out one of the faces in order to describe an object with the topology of an open box with five sides. The α_2 involutions of the darts along the rim of the opening would then refer to the dart itself, since there would be no adjacent face.

Reference is now made to FIG. 4, which illustrates in a UML diagram a data structure according to a preferred embodiment of the invention. In an object oriented programming language such as C++, the vertices, darts and faces would preferably be defined as classes/objects, while e.g. edges and volumes are defined only implicitly. Also, while some relationships are explicitly defined through the use of pointers, others are defined only indirectly and can be found by examining the value of several pointers. However, it would be possible to structure this differently. Other choices could be made for which definitions that should be made explicit and which should be implicit. Information that is presented here as included in one class of objects could be included in other classes of objects. Also, additional classes of objects that tie objects together could be included, so that e.g. pointers that here are presented as referencing other objects directly may do so only indirectly. It would even be possible to implement the data structure of the invention in a programming language that is not object oriented. The skilled person will realize that it is the exploitation of the G-map data structure in a mesh based geometry, rather than the expression of this data structure in any particular programming language, that represents one of the central ideas behind the invention.

For every G-map structure 401, there will be an arbitrary number of darts 402. As already described, the darts will reference each other by way of α-involutions. The α-involutions may be implemented as pointers in each dart-object in an object oriented programming language.

The geometry is embedded in the topology. According to a preferred embodiment, this is done by way of three additional objects: vertices 403, quads 404, and faces 405. A vertex 403 is the topological representation of a 0-orbit in the mesh. A vertex can be referred to by any number of darts, but each dart will refer to only one vertex. These references can be implemented as a pointer in each dart, pointing to the associated vertex, one or more pointers in each vertex, pointing to the associated darts, or both.

In the same way, a face is the topological representation of a single mesh cell. A face is surrounded by a loop of darts that are tied together by alternating α_0 and α_1-involutions. Again, each dart will be associated with only one face, while each face may be associated with an arbitrary number of darts. The pointers may again be implemented in the dart objects, in the face objects, or in both.

It should be noted that it is sufficient to include one pointer from a face object to one of the associated darts, since the remaining darts can be found as the referenced dart's 2-orbit. In the same way it is sufficient to include only one pointer from a vertex to a dart, since the remaining darts can be found as the referenced dart's 0-orbit.

Quads represent rectangular sections of each face. As will be described in greater detail with reference to FIG. 5, there can be an arbitrary number of quads to a face, but there will be only one vertex and two darts for each quad. According to a preferred embodiment of the invention, there is no explicit reference between vertices and quads (or indeed between any of the embedded objects), but each quad refers explicitly to two darts (actually to one, the other is found through an α_1-involution), and each dart refers to one quad.

According to a preferred embodiment of the invention, an embedding 406 is a class from which the various embedded objects inherit certain characteristics. In addition, the embedded objects will have additional characteristics depending on what kind of objects they are. Each vertex 403 will be associated with a variable holding the coordinates of a point in space 407, each quad 404 will be associated with a description of a surface patch 408, and each face will be associated with a variable holding the coordinates of a center point 409 of the face. According to a preferred embodiment it is the center point of a refined inner mesh that is stored as this center point 409, as will be described below.

The various variables and functions that describe geometry are preferably included as native to the embedded objects. However, they could also, as a matter of design choice, be located in separate objects that are pointed to by the embeddings, or in any other manner.

It should be noted that in the case where the G-map topology is used to describe a mesh that will be subdivided repeatedly and not associated with surface descriptions, the quad/patch and face/center point may be left out. One of the main advantages of the invention, i.e. the topology description that can be quickly navigated in order to implement local refinement or distortion, will still be achieved.

According to this exemplary embodiment, there are no objects that explicitly define edges. However, all the darts incident to an edge can be found as the 1-orbit of one of the darts incident to the edge. Also, according to this embodiment, there are no embedded objects that describe volumes, and there are no α-involutions above α_2. It should be noted, however, that the principles of the invention are not limited in this way, and that any number of dimensions could be included in principle.

Whether the data structure described is created as objects including pointers and algorithms, or through some other means, it will include a number of data structures that represent the various parts of a G-map mesh and the references, or associations, between these parts. The data structures will preferably be stored in a computer memory for access by a data processing system.

FIG. 5 shows in more detail how quads are derived from faces and associated with darts. In a first level of detail 501, the mesh of FIG. 2 is shown. In a second level of detail 502, the darts associated with one of the vertices of this mesh are shown, along with the quads associated with these darts. Finally, in a third level of detail 503, two darts are illustrated along with one quad, and the face of which it is part.

The quads of a face are, according to a preferred embodiment of the invention, found using midpoint refinement. This method is described with reference to FIG. 6 below. It should be noted that if the embedded geometry is described using repeated subdivision, quads are not necessary.

Each quad will be associated with one or both of the darts, as already described. It is not necessary to associate the quad with the face, since they will be linked through a common dart.

It should be noted that even if the face is linked to only one of the surrounding darts by a pointer in the face object, all the associated quads can be found by finding the 2-orbit of the dart pointed to by the face, and collecting all the quads pointed to by the darts in this orbit.

FIG. 6 illustrates in greater detail how the quads are constructed, and how a refined mesh is constructed from the first mesh.

The construction of quads described with reference to FIG. 5 and the subsequent association with a patch geometry, corresponds with what is known in the art as midpoint refinement. This is a method that is well known in the art, reference is made to Jörg Peters, Biquadratic C^1-surface splines over irregular meshes, Computer Aided Design Volume 27 Number 12, December 1995, pp. 895-903, 1995. Midpoint refinement is based on inserting points at the centroid of each mesh cell and the midpoint of each cell edge. This results in a refined structure of quadrilateral sub cells, and these are what we refer to as quads. Starting with the mesh 601, we get the patch structure 602.

From the original mesh 601, a refined mesh 603 is created. This is done by applying a mesh refinement algorithm to the original mesh. One such algorithm is illustrated in FIG. 6. For every quad in the original mesh a new vertex is found based on each original vertex and the two neighboring vertices. The coordinates (positions) to be associated with each vertex in the new mesh is then found as

$$V_0 = V + u^*V_U + v^*V_V + w^*V_U \Lambda V_V \qquad (1)$$

Where

u, v and w are coefficients that represent local shape parameters,

$V_U = V_R - V$

$V_V = V_L - V$

and V, V_L and V_R are the positions associated with the original vertex, and the vertices to the left and right of this in the original mesh, respectively. The vertex V_0 and the coefficients u, v, w, are preferably stored in the corresponding quad.

The resulting vertices of the refined mesh are used as control points when the shape of each patch is found. However, the refined mesh will not provide all the necessary control points for constructing a surface spline patch. This is illustrated in a detailed view 604 of the patching submesh. The Vertex V_0 gives the control point C_0, and other vertices of the refined mesh found in the same way but based on another set of vertices of the original mesh, yield the control points C_1, C_2, C_3, C_5, C_6, and C_7. Preferably, C_8 is the average of the points surrounding the face of the inner mesh, and according to a preferred embodiment of the invention, this is also the point that is stored as the center point of the face described above. In the same way, C_4 can be found as the average of the inner mesh vertices that have been based on one point V of the outer mesh.

The C-control points will be used to generate the Bezier control points that determine the actual shape of each patch. This will be described in greater detail with reference to FIG. 10 and FIG. 11.

Turning now to FIG. 7, local refinement of the model will be described. The description is based on the preferred embodiment using midpoint refinement in order to find a patch structure and creating surface spline patches. However, other embodiments are within the scope of the invention, e.g. using subdivision in order to create the smooth surfaces. Subdivision could e.g. be used independently for the coarse mesh and for the local refined mesh.

Again the mesh of FIG. 2 is shown at a first level of detail 701, while at a second level of detail 702, a pair of darts and an associated quad are shown. This quad is again subdivided into a second level of detail, and on this second level a new G-map is created. Hence, there is one G-map for each level.

The first level G-map, or level 0, comprises a dart_level_0 and a quad_level_0, with all the properties already described. The second level G-map, level 1, comprises a vertex, eight darts and four quads. The darts are associated with each other in pairs by α_1-involutions, and each such pair of darts on dart_level_1 span out one of the four quads on quad_level_1. In addition, each quad is associated with neighboring quads through α_2-involutions that tie together neighboring darts. Each level 1 quad includes three coefficients u, v, w and the coordinates of a refined level 1 inner mesh vertex V_0, found by the same method as that already described for level 0. The 3D point associated with the level 1 vertex will be taken from the V_0 vertex stored in the corresponding level 0 quad (quad_level_0).

The eight darts are associated with each other as described above. However, they will not necessarily include references to adjacent quads in the form of α_0 involutions, unless the refinement extends over neighboring quads, as described below. Consequently there may be no level 1 face that are part of the level 1 G-map.

Furthermore, the reason for creating the second level of detail that G-map level 1 represents could be the result of interaction between several objects. If e.g. an object is hit by another object somewhere inside a quad, causing a deformation, the vertex for this second level G-map, and possibly other vertices in the G-map, may be manipulated directly, and the various values will no longer be derived from the coarser level G-map.

Second, it should be noted that as long as the refinement does not extend beyond the original quad, none of the darts on dart_level_1 will be linked with other darts by α_0-involutions, and hence there will be no complete 2-orbit and therefore no face associated with any of these darts. However, if the refinement does extend beyond one quad, the second level G-map will include darts that are created in the adjacent

quads, and they will be linked with α_0-involutions across the quad boundaries, as shown in FIG. 8.

Finally it should be noted that it will be necessary to be able to navigate from a G-map on one level to a G-map on another level – finer or coarser. This is done by adding what will be referred to as γ-links as pointers from darts on one dart level (belonging to one G-map) to associated darts on another dart level (belonging to another G-map). Preferably, this is done by adding pointers to the dart objects. One pointer, $\gamma_{\_fine}$, points to a dart on the next finer level, and one pointer, $\gamma_{\_coarse}$, points to a dart on a coarser level. It is not, however, necessary to add these pointers to every dart object. When going from a coarser level to a finer level, it is sufficient to have a pointer from one of the two darts that are associated with the patch that is refined. If one dart does not contain such a reference, the other dart, found by applying an α_1-involution, will. Similarly, when going from a finer to a coarser level, it is sufficient to search the darts of the 0-orbit around a vertex in order to find the dart that contains a reference to the coarser level.

FIG. 8 shows local refinement similar to that of FIG. 7, but where the refinement involves two adjacent patches of the first, coarser mesh. It can be seen that the exact same procedure is followed, with the only difference that darts are linked across patch boundaries. It will also be noticed that there are still no complete α_2-orbits, so the finer mesh does not include any faces.

FIG. 9 is an illustration of the various cases that may occur when constructing the surface spline patches form control points found in an inner mesh. As described with reference to FIG. 6, the control points will be found as vertices in the inner mesh, or as averages of these, together with node valens (number fo faces around a node) and face valence (number of nodes around a face). However, in the case of local refinement, there will be a lot of border conditions where the G-map is incomplete, and where the inner mesh does not produce all these control points. However, since the aim is to construct smooth surfaces, the necessary information can be found from the coarser level G-maps. FIG. 9 illustrates five different cases, where case 0 represents the complete situation already described. Case 0 will also be the case for the first mesh level without any local refinement. The other cases represent situations where some of the control points cannot be found from the inner mesh of the G-map on the same level. The patch edges that follow the patch edges of the coarser level are in FIG. 9 shown as bold lines.

As mentioned above, the C-control points are not the Bezier control points (B-control points) used to actually compute the geometric shape of the patches. The B-control points are derived from the C-control points, and exactly how this is done may depend on the characteristics of the mesh where the patch is located.

FIG. 10 illustrates a bi-quadratic Bezier patch in a regular mesh area (node valence and face valence both equal 4). In this case it is sufficient with B-control points in each corner of the patch, halfway along each edge, and in the center of each patch. On the first mesh level of a totally regular mesh, all B-control points can be found from the C-control points. However, as already described, there will be cases where some of the C-control points are missing. Table 1 shows how the B-control points are derived from the C-control points in the cases illustrated in FIG. 9, as well as indicating which points are unavailable in this manner.

PATCHING REGULAR AREA QUADS

Control points	Case 0	Case 1	Case 2	Case 3	Case 4
B_{00}	C_4	as case 0	as case 0	as case 0	as case 0
B_{10}	$(C_0+C_5)/2$	as case 0	as case 0	as case 0	as case 0
B_{20}	$(C_0+C_5+C_6+C_7)/4$	--	as case 0	--	as case 0
B_{01}	$(C_0+C_3)/2$	as case 0	as case 0	as case 0	as case 0
B_{11}	C_0	as case 0	as case 0	as case 0	as case 0
B_{21}	$(C_0+C_7)/2$	--	as case 0	--	as case 0
B_{02}	$(C_0+C_1+C_2+C_3)/4$	--	--	as case 0	as case 0
B_{12}	$(C_0+C_1)/2$	--	--	as case 0	as case 0
B_{22}	C_8	--	--	--	--

Table 1.

As can bee seen from table 1, all points can be found only in case 0. In the remaining cases there is not enough information in the inner mesh constructed from the refined G-map. This means that it will be necessary to find this information from the coarser levels, since the transition from the level i surface to the level i+1 surface should be smooth. As already mentioned, the patch edges that follow the patch edges of the coarser level are in FIG. 9 shown as bold lines. These are the edges for which B-control points are not available directly from the C-control points on the same level. For these edges information will be found on the coarser levels.

FIG. 10 illustrates how B-control points on a refined level correspond with control points on the coarser level. When a patch is refined, four new patches are created (sub-patch 0 – 3), each with the same control point pattern as the coarser level patch. Going from a coarser to a more refined pattern of control points is well known from Bezier and spline theory. A detailed description can e.g. be found in Gerald Farin, Curves and Surfaces for CAGD, 5th edition, Morgan Kaufman Publishing, 2001.

In areas where the mesh is irregular it is necessary to introduce more B-control points. The reason for this is the need for enough free variables in order to make the surface tangentially continuous. It would be possible to generate surfaces using the

B-control points described above, but this would result in edges in the surface where a smooth surface is desired.

FIG. 11 shows an example of possible pattern of B-control points when the mesh is irregular, representing a preferred embodiment of the invention. In this case four cubic Bezier triangles are used to form a quad patch. The same principles apply in the sense that whenever possible, B-control points B^1_{ijk} are derived from the C-control points of the inner mesh, while when this is not possible, the necessary information is found on the coarser levels.

Tables 2 – 5 give the formulas for each B-control point, where

nv = node valence fv = face valence

Sub patch 1:

Control points	Case 0	Case 1	Case 2	Case 3	Case 4
B^1_{003}	$(C_0+C_1+C_2+C_3)/4$	--	--	as case 0	as case 0
B^1_{102}	$25/12(C_0+C_3)+(C_1+C_2)/12$	--	--	as case 0	as case 0
B^1_{201}	$(C_4+C_0+C_3)/3$	as case 0	as case 0	as case 0	as case 0
B^1_{300}	C_4	as case 0	as case 0	as case 0	as case 0
B^1_{012}	$5/12C_0+(C_1+C_3)/4+C_2/12$	--	--	as case 0	as case 0
B^1_{111}	$((2.0-\cos(\frac{2\pi}{nv}))/2(C_0+C_3)+(1+\cos(\frac{2\pi}{nv}))(C_0+C_1+C_2+C_3)/4+C_4)/6+C_0/3$	$(B^1_{102}+B^1_{201}+(C_0-C_3)/3)/2$	as case 1	as case 0	as case 0
B^1_{210}	$(C_4+C_0)/3+(C_3+C_5)/6$	as case 0	as case 0	as case 0	as case 0
B^1_{021}	$(((2.0-\cos(\frac{2\pi}{fv}))(C_0+C_1)/2+(2.0-\cos(\frac{2\pi}{nv}))(C_0+C_3)/2)+(2.0+\cos(\frac{2\pi}{fv})+\cos(\frac{2\pi}{nv}))(C_0+C_1+C_2+C_3)/4+C_8+C_4)/12+C_0/3$	$(B^1_{111}+B^4_{111})/2$	as case 1	as case 0	as case 0
B^1_{120}	$((2.0-\cos(\frac{2\pi}{nv}))(2C_0+C_3+C_5)/2+(1.0+\cos(\frac{2\pi}{nv}))(2C_0+C_1+C_2+C_3+C_5+C_6+C_7)/4)/12+C_0/3+C_4/6$	$(B^1_{111}+B^2_{111})/2$	as case 1	as case 1	as case 1
B^1_{030}	$(((2.0-\cos(\frac{2\pi}{fv}))(2C_0+C_1+C_7)/2+(2.0-\cos(\frac{2\pi}{fv}))(2C_0+C_3+C_5))/2+(2.0+\cos(\frac{2\pi}{fv})+\cos(\frac{2\pi}{nv}))(2C_0+C_1+C_2+C_3+C_5+C_6+C_7)/4)/24+C_0/3+(C_8+C_4)/12$	$(B^1_{111}+B^2_{111}+B^3_{111}+B^4_{111})/4$	as case 1	as case 1	as case 1

Sub patch 2:

Control points	Case 0	Case 1	Case 2	Case 3	Case 4
B^2_{003}	C_4	as case 0	as case 0	as case 0	as case 0
B^2_{102}	$(C_0+C_4+C_5)/3$	as case 0	as case 0	as case 0	as case 0
B^2_{201}	$5/12(C_0+C_5)+(C_6+C_7)/12$	--	as case 0	--	as case 0
B^2_{300}	$(C_0+C_5+C_6+C_7)/4$	--	as case 0	--	as case 0
B^2_{012}	B^1_{210}	as case 0	as case 0	as case 0	as case 0
B^2_{111}	$((2.0-\cos(\frac{2\pi}{nv}))/2(C_0+C_5)+$ $(1+\cos(\frac{2\pi}{nv}))(C_0+C_5+C_6+C_7)/4$ $+C_4)/6+C_0/3$	$(B^2_{102}+B^2_{201}+$ $(C_0-C_5)/3)/2$	as case 0	as case 1	as case 0
B^2_{210}	$(5/12C_0+(C_5+C_7)/4+C_6/12$	--	as case 0	--	as case 0
B^2_{021}	B^1_{120}	as case 0	as case 0	as case 0	as case 0
B^2_{120}	$(((2.0-\cos(\frac{2\pi}{nv}))(C_0+C_5)/2+$ $(2.0-\cos(\frac{2\pi}{fv}))(C_0+C_7)/2)+$ $(2.0+\cos(\frac{2\pi}{fv})+\cos(\frac{2\pi}{nv}))(C_0+C$ $_5+C_6+C_7)/4+C_4+C_8)/12+C_0/3$	$(B^3_{111}+B^2_{111})/$ 2	as case 1	as case 1	as case 1
B^2_{030}	B^1_{030}	as case 0	as case 0	as case 0	as case 0

Sub patch 3:

Control points	Case 0	Case 1	Case 2	Case 3	Case 4
B^3_{003}	B^2_{300}	as case 0	as case 0	as case 0	as case 0
B^3_{102}	$5/12(C_0+C_7)+(C_5+C_6)/12$	--	as case 0	--	as case 0
B^3_{201}	$(C_8+C_0+C_7)/3$	--	--	--	--
B^3_{300}	C_8	--	--	--	--
B^3_{012}	B^2_{210}	as case 0	as case 0	as case 0	as case 0
B^3_{111}	$((2.0-\cos(\frac{2\pi}{fv}))(C_0+C_7)/2+(1.0+\cos(\frac{2\pi}{fv}))(C_0+C_5+C_6+C_7)/4+C_8)/6+C_0/3$	--	$(B^3_{102}+B^3_{201}+(C_0-C_7)/3)/2$	--	as case 2
B^3_{210}	$(C_8+C_0)/3+(C_7+C_1)/6$	--	--	--	--
B^3_{021}	B^2_{120}	as case 0	as case 0	as case 0	as case 0
B^3_{120}	$((2.0-\cos(\frac{2\pi}{fv}))(C_0+C_7+C_0+C_1)/2+(1.0+\cos(\frac{2\pi}{fv}))((2C_0+C_5+C_6+C_7+C_1+C_2+C_3)/4))/12+C_0/3+C_8/6$	$(B^3_{111}+B^4_{111})/2$	as case 1	as case 1	as case 1
B^3_{030}	B^1_{030}	as case 0	as case 0	as case 0	as case 0

Sub patch 4:

Control points	Case 0	Case 1	Case 2	Case 3	Case 4
B^4_{003}	B^3_{300}	as case 0	as case 0	as case 0	as case 0
B^4_{102}	$(C_8+C_0+C_1)/3$	--	--	--	--
B^4_{201}	$5/12(C_0+C_1)+(C_2+C_3)/12$	--	--	as case 0	as case 0
B^4_{300}	B^1_{003}	as case 0	as case 0	as case 0	as case 0
B^4_{012}	B^3_{210}	as case 0	as case 0	as case 0	as case 0
B^4_{111}	$((2.0-\cos(\frac{2\pi}{fv}))(C_0+C_1)/2+(1.0+\cos(\frac{2\pi}{fv}))(C_0+C_1+C_2+C_3)/4+C_8)/6+C_0/3$	--	--	$(B^4_{102}+B^4_{201}+(C_0-C_1)/3)/2$	as case 3
B^4_{210}	B^1_{012}	as case 0	as case 0	as case 0	as case 0
B^4_{021}	B^3_{120}	as case 0	as case 0	as case 0	as case 0
B^4_{120}	B^1_{021}	as case 0	as case 0	as case 0	as case 0
B^4_{030}	B^1_{030}	as case 0	as case 0	as case 0	as case 0

γ-links, it is not necessary to search through all the information available in order to find the control points. It is sufficient to travel through the refined G-map structure until the relevant γ-link is found, and travel along the relevant edges of this coarser G-map in order to find the right data.

Turning now to FIG. 12, a data structure corresponding with that in FIG. 4 is shown, illustrating how the various levels are linked. It can be seen that the links between objects on the same level (the same G-map) are the same as in FIG. 4, while additional γ-links to finer and coarser G-map levels are also included. For the finest and the coarsest level, the $\gamma_{\_finer}$ and $\gamma_{\_coarser}$ links, respectively, will be null pointers.

FIG. 13 illustrates the input data that is used in order to create a single level G-map in accordance with the invention. The nodes are numbered $N_0 \ldots N_6$, and for each node a coordinate is given. For the sake of simplicity, the example is two dimensional, giving only two coordinates for each node. In the same way the faces are numbered $F_0 \ldots F_2$, and for each face the number of associated nodes are given, followed by the identifying number of each such node. Finally the quads are listed, but only as parameters to be used when operating on the input data. For each quad the associated pair of node and face is given. Following that constants that represent blend ratios are given. These values are used when computing the vertices of the inner mesh. Finally the level of refinement is given. In this case all the quads belong to level 0, since no refinement has taken place.

FIG. 14 shows a similar example, but here the model has been refined with an additional level. The nodes and faces are the same, and the list of quads for level 0 has been changed only in that there is an indication that this level has been refined on level 1. Following this is a list of quads for this new level. In this case, since it is quite possible that there will not be any complete faces on this level, and indeed there are none in this example, the quads do not refer to faces and nodes, but to the coarser level quad of which it is a refinement, and simply a number that is defined according to the sequence in which the quads can be found when following a 0-orbit around the refinement point starting with the dart with which the γ-links are associated. Other ways of enumerating the quads of a refined G-map level are possible within the scope of the invention.

Finally, turning to FIG. 15, an example is given illustrating a screenshot of a 3D object that is being refined. In FIG 15a a 3D object is illustrated along with the outer mesh G-map and the refined inner mesh from which the control points are computed. FIG. 15b illustrates the same 3D object, but here the various patches corresponding with quads are indicated. In FIG. 15c one of the quads has been refined, and the resulting four patches on the refined level and the level 1 refined inner mesh are shown. The level 1 outer mesh G-map only consists of a vertex in

the level 0 refined inner mesh, and cannot be seen in the drawing. In FIG. 15d a similar refinement has been performed on a neighboring quad, and now the level 1 outer mesh G-map can be seen in the drawing as a bold line along the refined inner mesh of level 0.

FIG. 15e illustrates a local deformation performed after the refinements have been performed, and now the outer mesh G-map of level 1 is shown as a closed bold line. FIG. 15f shows the pattern of patches of the refined and deformed model. It can be seen that one of the second level patches has been further refined by a third level G-map as indicated by the patch pattern on the local deformation.

The invention has been described in terms of a method for describing a data structure containing topological and geometric information. The method is primarily intended to be implemented on a computer system with the necessary hardware component to process and store this information and render it in the form of graphics on a display unit.

The invention may be implemented as computer software instructions stored on a computer readable medium, such as a hard drive in a computer system or in other memory on such a system, or on other computer readable medium such as a CD-ROM, a disk or any other kind of magnetic or magneto-optical storage medium. The instructions comprising the computer program may also be carried on a propagated signal.